METHOD AND DEVICE FOR BUILDING A VARIABLE-LENGTH ERROR-CORRECTING CODE

FIELD OF THE INVENTION

The present invention relates to a method of building a variable length error code, said method comprising the steps of :

(1) initializing the needed parameters : minimum and maximum length of codewords L_1 and L_{max} respectively, free distance d_{free} between each codeword (said distance d_{free} being for a VLEC code C the minimum Hamming distance in the set of all arbitrary extended codes), required number of codewords S ;

(2) generating a fixed length code C of length L_1 and minimal distance b_{min}, with $b_{min} = \min \{b_k ; k = 1, 2, \ldots\ldots, R\}$, b_k = the distance associated to the codeword length L_k of code C and defined as the minimum Hamming distance between all codewords of C with length L_k, and R = the number of different codeword lengths in C, said generating step creating a set W of n-bit long words distant of d ;

(3) listing and storing in the set W all the possible L_1 – tuples at the distance of d_{min} from the codewords of C (said distance d_{min} for a VLEC code C being the minimum value of all the diverging distances between all possible couples of different-length codewords of C), and, if said set W is not empty, doubling the number of words in W by affixing at the end of all words one extra bit, said storing step therefore replacing the set W by a new one having twice more words than the previous one and the length of each one of these words being $L_1 + 1$;

(4) deleting all the words of the set W that do not satisfy the c_{min} distance with all codewords of C, said distance c_{min} being the minimum converging distance of the code C ;

(5) in the case where no word is found or the maximum number of bits is reached, reducing the constraint of distance for finding more words ;

(6) controlling that all words of the set W are distant of b_{min}, the found words being then added to the code C ;

(7) if the required number of codewords has not been reached, repeating the steps (1) to (6) until the method finds either no further possibility to continue or the required number of codewords ;

(8) if the number of codewords of C is greater than S, calculating on the basis of the structure of the VLEC code, the average length AL obtained by weighting

each codeword length with the probability of the source, said AL becoming the AL_{min}, if it is lower than AL_{min}, with AL_{min} = the minimum value of AL, and the corresponding code structure being kept in memory.

The invention also relates to a corresponding device.

BACKGROUND OF THE INVENTION

A classical communication chain, illustrated in Fig.1, comprises, for coding the signals coming from a source S, a source coder 1 (SCOD) followed by a channel coder 2 (CCOD) and, after the transmission of the coded signals thus obtained through a channel 3, a channel decoder 4 (CDEC) and a source decoder 5 (SDEC). The decoded signals are intended to be sent towards a receiver. Variable-length codes (VLC) are classically used in source coding for their compression capabilities, and the associated channel coding techniques combat the effects of the real transmission channel (such as fading, noise, etc.). However, since source coding is intended to remove redundancy and channel coding to re-introduce it, it has been investigated how to efficiently coordinate these techniques in order to improve the overall system while keeping the complexity at an acceptable level.

Among the solutions proposed in such an approach, the variable-length error correcting (VLEC) codes present the advantage to be variable-length while providing error correction capabilities, but building these codes is rather time consuming for short alphabets (and become even prohibitive for higher length alphabets sources), and the construction complexity is also a drawback, as it will be seen.

First, some definitions and properties of the classical VLC must be recalled. A code C is a set of S codewords $\{c_1, c_2, c_3, \ldots, c_i, \ldots c_S\}$, for each of which a length $\ell_i = |c_i|$ is defined, with $\ell_1 \leq \ell_2 \leq \ell_3 \leq \ldots \leq \ell_i \leq \ldots \leq \ell_S$ without any loss of generality. The number of different codeword lengths in the code C is called R, with obviously $R \leq S$, and these lengths are denoted as $L_1, L_2, L_3, \ldots, L_i, \ldots L_R$, with $L_1 < L_2 < L_3 < \ldots < L_R$. A variable-length code, or VLC, is then the structure denoted by $(s_1@ L_1, s_2@ L_2, s_3@ L_3, \ldots, s_R@ L_R)$, which corresponds to s_1 codewords of length L_1, s_2 codewords of length L_2, s_3 codewords of length L_3, ..., and s_R codewords of length L_R. When using a VLC, the compression efficiency, for a given source, is related to the number of bits necessary to transmit symbols from said source. The measure used to estimate this efficiency is often the average length AL

of the code, i.e. the average number of bits needed to transmit a word, and said average length is given, when each symbol a_i is mapped to the codeword c_i, by the following relation (1) :

$$AL = \sum_{i=1}^{i=s} \ell_i \, . \, P(a_i) \qquad (1)$$

which is equivalent to the relation (2) :

$$AL = \sum_{i=1}^{R} L_i \, . \, \left(\sum_{j=r(i)+1}^{j=r(i+1)} P(a_j) \right) \qquad (2)$$

where, for a data source A, the S source symbols are denoted by $\{a_1, a_2, a_3, \ldots, a_s\}$ and $P(a_i)$ is the respective probability of occurrence of each of these symbols, with $\sum P(a_i) = 1$ (from i = 1 to i = S). If AL_{min} denotes the minimal value for the average length AL, it is easy to see that when AL_{min} is reached, the symbols are indexed in such a way that $P(a_1) \geq P(a_2) \geq P(a_3) \geq .. \geq P(a_i) \geq \ldots P(a_s)$. In order to encode the data in such a way that the receiver can decode the coded information, the VLC must satisfy the following properties : to be non-singular (all the codewords are distinct, i.e. no more than one source symbol is allocated to one codeword) and to be uniquely decodable (i.e. it is possible to map any string of codewords unambiguously back to the correct source symbols, without any error).

An introduction and a presentation of different distances that are useful when reviewing some general properties of the VLC codes will then help to recall the notion of error-correcting property used in the VLEC code theory :

(a) Hamming weight and distance : if w is a word of length n with $w = (w_1, w_2, \ldots, w_n)$, the Hamming weight of w, or simply weight, is the number W(w) of non-zero symbols in w :

$$W(w) = \sum_{i=1}^{i=n} \frac{w_i}{\|w_i\|} \qquad (3)$$

and, if w_1 and w_2 are two words of equal length n with $w_i = (w_{i1}, w_{i2}, w_{i3}, \ldots, w_{in})$ and i = 1 or 2, the Hamming distance (or, simply, distance) between w_1 and w_2 is the number of positions in which w_1 and w_2 differ (for example, for the binary case, it is easy to see that :

$$H(w_1, w_2) = W(w_1 + w_2) \qquad (4)$$

where the addition is modulo-2). However, the Hamming distance is by definition restricted to fixed-length codes, and other definitions will be defined before considering VLEC codes.

(b) let $f_i = w_1^i\ w_2^i \ldots. w_n^i$ be a concatenation of n words of a VLEC code C, then the set $F_N = \{f_i : |f_i| = N\}$ is called the extended code of C of order N.

(c) minimum block distance and overall minimum block distance : the minimum block distance b_k associated to the codeword length L_k of a VLEC code C is defined as the minimum Hamming distance between all distinct codewords of C with the same length L_k :

$$b_k = \min \{H(c_i, c_j) : c_i, c_j \in C, i \neq j, |c_i| = |c_j| = L_k\} \text{ for } k = 1,\ldots, R \qquad (5)$$

and the overall minimum block distance b_{min} of said VLEC code C, which is the minimum block distance value for every possible length L_k, is defined by :

$$b_{min} = \min \{b_k : k = 1,\ldots R\} \qquad (6)$$

(d) diverging distance and minimum diverging distance : the diverging distance between two codewords of different length $c_i = x_{i_1} x_{i_2} \ldots. x_{i_{\ell_i}}$ and $c_j = x_{j1} x_{j2} \ldots. x_{j_{\ell_j}}$ of a VLEC code C, where $c_i, c_j \in C$, $\ell_i = |c_j|$ and $\ell_j = |c_j|$ with $\ell_i > \ell_j$, is defined by :

$$D(c_i, c_j) = H(x_{i_1} x_{i_2} \ldots. x_{i_{\ell_i}}, x_{j_1} x_{j_2} \ldots. \ldots x_{j_{\ell_j}}) \qquad (7)$$

i.e. it is also the Hamming distance between a ℓ_j - length codeword and the ℓ_j - length prefix of a longer codeword, and the minimum diverging distance d_{min} of said VLEC code C is the minimum value of all the diverging distances between all possible couples of codewords of C of unequal length :

$$d_{min} = \min \{ D(c_i, c_j) : c_i, c_j \in C, |c_i| \neq |c_j| \} \qquad (8)$$

(e) converging distance and minimum converging distance : the converging distance between two codewords of different length $c_i = x_{i_1} x_{i_2} \ldots x_{i_{\ell_i}}$ and $c_j = x_{j1} x_{j2} \ldots. x_{j_{\ell_j}}$ of a VLEC code C, where $|c_i| = \ell_i > |c_j| = \ell_j$, is defined by :

$$C(c_i, c_j) = H(x_{i_{\ell_i - \ell_j + 1}} x_{i_{\ell_i - \ell_j + 2}} \ldots x_{i_{\ell_i}}, x_{j_1} x_{j_2} \ldots x_{j_{\ell_j}}) \qquad (9)$$

i.e. it is also the Hamming distance between a ℓ_j - length codeword and the ℓ_j - length suffix of a longer codeword, and the minimum converging distance of said VLEC code C is the minimum value of all the converging distances between all possible couples of C of unequal length :

$$c_{min} = \min \{C(c_i, c_j) : c_i, c_j \in C, |c_i| \neq |c_j|\} \qquad (10)$$

(f) free distance : the free distance d_{free} of a code is the minimum Hamming distance in the set of all arbitrary long paths that diverge from some common state S_i and converge again in another common state S_j, with $j > i$:

$$d_{free} = \min \{H(f_i, f_j) : f_i, f_j \in F_N, N = 1, 2, \ldots, \infty\} \qquad (11)$$

Following the structure model used for a VLC, it is therefore possible to describe the structure of the VLEC code C by the notation :

$$S_1 @ L_1, b_1 ; S_2 @ L_2, b_2 ; \ldots ; S_R @ L_R, b_R ; d_{min}, c_{min} \qquad (12)$$

where there are s_i codewords of length L_i with minimum block distance b_i, for all $i = 1, 2, \ldots R$, (it is recalled that R is the number of different codeword lengths) and minimum diverging and converging distances d_{min} and c_{min}. The most important parameter of a VLEC code is its free distance d_{free}, which influences greatly its performance in terms of error-correcting capabilities, and it can be shown that the free distance of a VLEC code is bounded by :

$$d_{free} \geq \min (b_{min}, d_{min} + c_{min}) \qquad (13)$$

These definitions being recalled, the state-of-the-art in VLEC codes construction will be now described more easily. The first types of VLEC codes, called α -prompt codes and introduced in 1974, and an extension of this family, called $\alpha_{t_1,t_2,\ldots,t_R}$ -prompt codes, have both the same essential property : if one denotes by $\alpha(c_i)$ the set of words that are closer to c_i than to any codeword c_j, with $j \neq i$, no sequence in $\alpha(c_i)$ is a prefix of a sequence in another $\alpha(c_i)$. The construction of these codes is very simple, and the construction algorithm is adjustable by the number of codewords at each length, which makes possible to find the best prompt code for a given source and a given d_{free}. However, this best code performs poorly in terms of compression performance.

A more recent construction, allowing the construction of a VLEC code from the generator matrix of a fixed-length linear block code, was proposed in the document "Variable-length error-correcting codes" by V.Buttigieg, Ph.D.Thesis, University of Manchester, England, 1995. Called code-anticode construction, this algorithm relies on line combinations and column permutations to form an anticode at the rightmost column. Once the code-anticode generator matrix is obtained, the VLEC code is simply obtained by a matrix multiplication.

This technique has however several drawbacks. First, there is no explicit method to find the needed line combinations and column permutations to obtain the anticode. Moreover, the construction does not take into account the source statistics and, consequently, often reveals itself sub-optimal (one can find a code with smaller average length by a post-processing on the VLEC code). In the same document, the author has then proposed an improved method, called Heuristic method, that is based on a computer search for building a VLEC code giving the better known compression rate for a specified source and a given protection against errors, i.e. a code C with specified overall minimum block, diverging and converging distances (and hence a minimum value for d_{free}) and with codeword lengths matched to the source statistics so as to obtain a minimum average codeword length for the chosen free distance and the specified source (in practice, one takes : $b_{min} = d_{min} + c_{min} = d_{free}$, and : $d_{min} = [d_{free}/2]$.

The main steps of this Heuristic method, which uses the following parameters : minimum length L_1 of codewords, maximum length L_{max} of codewords, free distance d_{free} between each codeword, number S of codewords required, are now described with reference to the flowcharts of Figs.2 to 4.

To start the computer search ("Start"), all the needed parameters must be first specified : L_1 (the minimum codeword length, which must be at least equal to or greater than the minimum diverging distance required), L_{max} (the maximum codeword length), the different distances between codewords (d_{free}, b_{min}, d_{min}, c_{min}), and S (the number of codewords required by the given source), and some relations are set when choosing these parameters :

$L_\ell \geq d_{min}$

$b_{min} = d_{free}$

$d_{min} + c_{min} = d_{free}$

The first phase of the algorithm, referenced 11, is then performed : it consists in the generation of a fixed length code (put initially in C) of length L_1 and minimal distance b_{min}, with a maximum number of codewords. This phase is in fact an initialization, performed for instance by means of an algorithm such as the greedy algorithm (GA), presented in Fig.5, or the majority voting algorithm (MVA), presented in Fig.7, or a new proposed variation, denoted by GAS (Greedy Algorithm by Step), which consists of a variation of the two above mentioned ones. The GAS consists in the search method used in the GA, where instead of deleting half of the

codewords, only the last codeword of the group is deleted. These two algorithms are useful to create a set W of n-bit long words distant of d (in practice, it may be noted that the MVA finds more words than the GA, but it asks too much time for only a small improvement of the compression capacity, as shown in the tables of Figs.6 and 8, which compare, respectively for the GA and for the MVA, the best code structures obtained with different values of d_{free} for the 26-symbol English source defined in the table of Fig.9.

The second phase of the algorithm, corresponding to the elements referenced 21 to 24 (21+22 = operation "A0" ; 23+24 = operation "A2") in Fig.2, consists in listing and storing (step 21) in a set called W all the possible L_1 - tuples at the distance of d_{min} from the codewords in C. If $d_{min} \geq b_{min}$, then W is empty. If this set W of all the words satisfying the minimum diverging distance to the current code is not empty (reply NO to the test 22 : $|W| = 0$?), the number of words in W is doubled by increasing the length of the words by one bit by affixing first a "0" and then a "1" to the rightmost position of all the words in W (step 24), except if the maximum number of bits is exceeded (reply YES to the test 23). At the output of said step 24, this modified set W has twice more words than the previous W, and the length of each one is $L_1 + 1$.

The third phase of the algorithm, corresponding to the elements 31 to 35 (= operation "A3" in Fig.2), consists in deleting (step 31) all the words of set W that do not satisfy the c_{min} distance (minimum converging distance) with all the codewords of C (i.e. in keeping and storing in a new W only the words which satisfy said minimum converging distance, the other ones being discarded). At this point, the new set W is a set of words which, when compared to the codewords of C, satisfy the required minimum diverging and converging distances (both d_{min} and c_{min} distances) with the codewords of C. If that new set W is not empty (reply NO to the test 32 : $|W| = 0$?) one selects in W (step 33) the maximum number of words to satisfy the minimum block distance, in order to ensure that all the words of the set W, being of the same length, have a minimum distance at least equal to b_{min}. At the end of this step 33, realized with the GA or the MVA (note that in this case, the initial set used for the GA or the MVA is the current W and not a n-tuples set), the words thus obtained are added (step 34) to the codewords already in C.

If no word is found (i.e. W is empty) at the end of the step 21 (reply YES to the test 22 : $|W| = 0$?) or if the maximum number of bits is reached or exceeded (reply YES to the test 23), one enters the fourth phase of the algorithm (steps 41 to 46, illustrated in Fig.3 and also designated by the operation "A1" in said figure), which is used in order to unjam the process by inserting more liberty of choice, more particularly by affixing to all words in W extra bits (several bits at the same time) such that the new group contains more bits than the old one. If there are enough codewords in the last group (successive tests 41 and 42, for verifying the number of codewords in the last group, and if there are previous groups), some of them are deleted from said group (as described above), such deletions allowing to reduce the distance constraint and to find more codewords than before. As a matter of fact, the classical Heuristic method thus described begins with the maximum of codewords with the short length, maps them with the high probability symbols and tries to obtain a good compression rate, but sometimes the size of the small lengths sets are incompatible with the required number of codewords S. In this optic, easing a few codewords provides more freedom degrees and allows to reach a position where the initial requirements on distance and number of symbols for the code can be met. This deletion process is repeated until it remains a maximum of one codeword for each length. If W is empty at the end of the step 31 (reply YES to the test 32 : $|W| = 0$?), the steps 23, 24, 31, 32 are repeated. If the required number of codewords has not been reached (reply NO to the test 35 provided at the end of this third phase), the steps 21 to 24 and 31 to 35 must be repeated until said steps find that either there are no further possible words to be found or the required number of codewords is reached.

If said required number of codewords has been reached (i.e. the number of codewords of C is equal to or greater than S (reply YES to the test 35), the structure of the VLEC code thus obtained is used in a fifth part, including the steps 51 to 56 (illustrated in Fig.4, and also designated by the operation "A4" in said figure), in order to calculate the average length AL. This is done by weighting each codeword length with the probability of the source, and comparing it to the current best one. If said average length AL of this VLEC code is lower than the minimized value of AL ($= AL_{min}$), this AL becomes the AL_{min}, and this new AL value and the corresponding code structure are kept in the memory (step 51). These steps 51 and following (fifth part ; operation "A4") allow to come back, within the algorithm, towards previous

groups, while the other phases of said algorithm are always performed on the current group. The stepsize for such a feedback operation is one, i.e. this feedback action can be considered as exhaustive.

To continue this search of the best VLEC code, it is necessary to avoid keeping the same structure, which would lead to a loop in the algorithm. The last added group of the current code is deleted (steps 52, 53), the deletion of shorter length codewords allowing to find more longer length codewords (test 54 : number of codewords in group greater than 1 ?), and some codewords (half the amount for the GVA ; the "best" one for the MVA) of the previous group are deleted (step 55), in order to re-loop (step 56) the algorithm at the beginning of the step 21 (see Fig.2) and find different VLEC structures (the number of deleted codewords depends on which method is used for selecting the words : if the GA method is used and one wants to obtain a linear code, it is necessary to delete half of the codewords, while with the MVA method only one codeword, the best one, is deleted, i.e. the one that allows to find the more codewords in the next group).

However, the Heuristic method thus described often considers very unlikely code structures or proceeds with such a care (in order not to miss anything) that a great complexity is observed in the implementation of said method, which moreover is rather time consuming and can thus become prohibitive. It has therefore been proposed, in a European patent application filed on October 23, 2002, with the filing number 02292624.0 (PHFR020110), an improved construction method with which it is possible to gain in complexity by avoiding these drawbacks, said method of building a variable length error code comprising, more precisely, the steps of :

(1) initializing the needed parameters : minimum and maximum length of codewords L_1 and L_{max} respectively, free distance d_{free} between each codeword (said distance d_{free} being for a VLEC code C the minimum Hamming distance in the set of all arbitrary extended codes), required number of codewords S ;

(2) generating (step 11) a fixed length code C of length L_1 and minimal distance b_{min}, with $b_{min} = \min \{b_k ; k = 1, 2, \ldots\ldots, R\}$, b_k = the distance associated to the codeword length L_k of code C and defined as the minimum Hamming distance between all codewords of C with length L_k, and R = the number of different codeword lengths in C, said generating step 11 creating a set W of n-bit long words distant of d ;

(3) listing and storing (step 21) in the set W all the possible L_l – tuples at the distance of d_{min} from the codewords of C (said distance d_{min} for a VLEC code C being the minimum value of all the diverging distances between all possible couples of different-length codewords of C), and, if said set W is not empty, doubling the number of words in W by affixing at the end of all words one extra bit, said storing step therefore replacing the set W by a new one having twice more words than the previous one and the length of each one of these words being $L_l + 1$;

(4) deleting (step 31) all the words of the set W that do not satisfy the c_{min} distance with all codewords of C, said distance c_{min} being the minimum converging distance of the code C ;

(5) in the case where no word is found or the maximum number of bits is reached, reducing (step 41) the constraint of distance for finding more words ;

(6) controlling that all words of the set W are distant of b_{min}, the found words being then added to the code C (step 34) ;

(7) if (step 35) the required number of codewords has not been reached, repeating the steps (1) to (6) (i.e. the steps 21 to 35) until the method finds either no further possibility to continue or the required number of codewords ;

(8) if the number of codewords of C is greater than S, calculating (operation A4), on the basis of the structure of the VLEC code, the average length AL obtained by weighting each codeword length with the probability of the source, said AL becoming the AL_{min}, if it is lower than AL_{min}, with AL_{min} = the minimum value of AL, and the corresponding code structure being kept in memory ;

said building method being moreover such that at most one bit is added at the end of each word of the set W.

Simulations show that, with the classical Heuristic method, almost none of the obtained best codes has a hole (i.e. a length jump in its structure length). It is then considered, in the previously cited European patent application, that most good codes do not have jump of length and, therefore, that the set of examined VLEC codes can be reduced accordingly (which reduces the simulation time and the complexity of implementation of the method, without modifying much the AL). Following this hypothesis, the method has been, according to said European patent application, modified by avoiding to add more than one bit at the end of each word of the set W. The corresponding implementation (improved Heuristic construction method, also called "noHole optimization" method) is illustrated in Figs 10 and 11,

which show the two parts of a flowchart corresponding to said method (the elements that are identical to the ones observed in Figs.2 to 4 being designated with the same references). With respect to the flowchart of Figs.2 to 4, the parts that, with respect to the classical Heuristic technique, are useless for the implementation of the improved method have been cancelled :

(a) if W is empty at the end of the step 31 (reply YES to the test 32 : $|W| = 0$?), the next phase is now (see Fig.10) not the repetition of the steps (23, 24, 31, 32), but, according to said "noHole" method, the establishment (in place of said repetition) of a direct connection 91 towards the input of the circuit carrying out the operation 55 (deletion of some codewords, or of the best one, before a repetition of the steps 21 to 24 and 31 to 35), said operation 55 being then, as previously, followed by the operations 21 and following.

(b) the fourth phase of the method is now reduced to one step, the operation 41, which is the test "Number of codewords in last group = 1 ?". If the reply is NO, a direct link is established with the input of the step 55 (connection 91), in view of carrying out said operation 55, and then the operations 21 and following. If the reply is YES, a connection 92 is established with the input of the set of operations 52 to 54.

The results thus obtained are presented in the table of Fig.12 for the 26 symbol English source when using the GAS method for selecting codewords. It can be seen, when comparing with results presented in Fig.13, that although the result is not completely optimal for $d_{free} = 3$ (the code structure has a hole at length $L = 11$), the AL rise is really acceptable when one considers that there is both strictly no degradation for the other d_{free} values and a gain of time between 2,5 and 4. The same remarks can be applied when comparing the present solution with the ones obtained in Fig.7, where the MVA complexity effect is clear. Similarly, applying the noHole optimisation with the GA method for selecting codewords leads to a time gain at the only expense of a slight AL rise for $d_{free}=3$. Finally, Fig.5 shows on the other hand that the current solution offers better AL for an acceptable gain of time, the noHole optimisation compensating almost entirely the complexity induced by the GAS.

However, with the method thus described in said European patent application, there are cases where there are too many small length codewords in the generated VLEC code. It has then been proposed, in another European patent application filed on March 11, 2003, with the filing number 03290604.2 (PHFR030026), another

improved building method according to which the group deletion is not only performed with the last obtained codewords group, but more generally with groups up to a given length value group, in order to make possible to go back directly, and therefore very quickly, to smaller lengths, i.e. to skip many algorithm steps in cases where there are too many small length codewords. More precisely, denoting by L_s (with s for : skip) the length to which the algorithm will skip back to in the codeword deletion stage, it has been proposed to skip parts of the original Heuristic algorithm by carefully jumping to lower lengths when looking for codewords to be deleted (however, when the considered codewords group length L is smaller than a preset value L_s, it is obviously better to apply the previous method, and the deletion is then done within the group of length L). The length comprised between L_1 and L_s are consequently called "free lengths", i.e. lengths with a freedom degree, as they are decremented one by one in the search process (when the number of free lengths grows up, the simulation time also increased, exponentially). This method, called "L_s optimization adding", is depicted in the flowchart formed by the association of Fig.10 (unchanged part of the previous method, the so-called noHole optimization method) and Fig.14 (modified part of the noHole optimization method).

Said Fig.14 is adapted from Fig.11 according to the following indications. The last added group of the current code is deleted, but only if (test 61) the codeword length of this previous group is (reply NO to the test 61) lower than or equal to L_s (the steps that follow the test 61 are then the same as previously : steps 53, 54, and 55 or 52 at the output of the test 54). If said codeword length is greater than L_s (reply YES to the test 61), an additional step 62 is provided for going to group with L_s-bit long codewords and deleting all groups with more than L_s bits. At the output of the step 62, the same steps 54, 55 as previously are provided. In practice, simulation results show that good compression rates can be obtained for $L_s < L(max)$, where L(max) is the maximal authorized codeword length (it can be noted that increasing the value of L_s results in an improvement of the AL value until a constant floor – the best value – is reached, and this behaviour then suggests a possible dynamic choice of L_s, starting with $L_s = L_1$ and incrementing it until said floor reached). However, when updating the L_s parameter for a new search, it appears that no advantage is taken from previous researches.

SUMMARY OF THE INVENTION

It is therefore an object of the invention to propose an improved construction method with which this drawback is avoided and it is possible to reach higher L_s levels more quickly, in order to find better compression gains for an acceptable computation time.

To this end, the invention relates to a method such as defined in the introductory part of the description and which is moreover characterized in that the deletion is done not only in the last obtained group but also in the group of a given length value and, denoting by Ls the length of the code to which the method skips back at the end of the deletion step, the beginning of the best VLEC structure of each Ls is kept in memory and re-used within the next search for Ls' = Ls +1.

According to a possible improved implementation, the invention relates to a similar method (i.e. such as defined in the introductory part of the description), but which is now preferably characterized in that at most one bit is added at the end of each word of the set W, the deletion is done not only in the last obtained group but also in the group of a given length value, and, denoting by Ls the length of the code to which the method skips back at the end of the deletion step, the beginning of the best VLEC structure of each Ls is kept in memory and re-used within the next search for Ls' = Ls +1.

It is also an object of the invention to propose a device for carrying out said construction method.

To this end, the invention relates to a device for carrying out a variable length error code building method according to anyone of the two solutions thus proposed.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described, by way of example, with reference to the accompanying drawings in which :

- Fig.1 depicts a conventional communication channel ;

- Figs 2 to 4 are the three parts of a single flowchart illustrating the main steps of a conventional method used for building a VLEC code (and called Heuristic method) ;

- Fig.5 illustrates an algorithm (called greedy algorithm, or GA) used for the initialization of the method of Figs 2 to 4, and Fig.6 is a table giving various VLEC

codes for a source constructed with the Heuristic construction using said algorithm of Fig.5 ;

- Fig.7 illustrates another algorithm (called majority voting algorithm, or MVA) used for the initialization of the method of Figs 2 to 4, and Fig.8 is another table giving various VLEC codes for a source constructed with the Heuristic construction using said algorithm of Fig.7 ;

- Fig.9 is a table giving for the 26-symbol English source the correspondence between the source symbol and its probability ;

- Figs 10 and 11 are the two parts of a single flowchart illustrating an implementation of an improvement of the conventional method illustrated in Figs 2 to 4 ;

- Fig.12 is another table giving various VLEC codes for the same 26 symbol English source as considered in the tables of Figs 6 and 8 and using the GAS ;

- Fig.13 is another table giving various VLEC codes for the same source as in Fig.12 and using both the GAS previously mentioned and the building method according to the improvement illustrated in Figs 10 and 11 ;

- Fig.14 shows the modification of the part of flowchart of Fig.11 according to another improvement of the conventional method illustrated in Figs 2 to 4 ;

- Fig.15 is a table illustrating the results obtained for the 26-symbol English source when the previous method of Fig.14 ("Ls optimization" method) is carried out ;

- Fig.16 is a table illustrating similarly the results obtained for another source.

DETAILED DESCRIPTION OF THE INVENTION

Considering the results of some simulations made on the basis of the above-described "Ls optimization" method, it appears, as indicated above, that, when updating the Ls parameter for a new search, no advantage is taken from eventual previous researches. According to the invention, it is therefore proposed to try to establish a semi-recursive way to reach higher Ls levels quickly, in order to find better compression gains for an acceptable computation time.

More precisely, it is proposed to keep in memory the beginning of the best VLEC structure of each Ls, and to re-use it within the search with the next Ls' = Ls +1 value. As Ls rises, the size of the kept beginning increases accordingly, in order to avoid a resulting increase of the free length that would exponentially impact on

the computation time. In fact, simulations show that, when Ls increases, the beginning of each code remains constant for more and more lengths, which justifies to use the pre-computed information : whereas the previous method previously tested all combinations for length in [L1, Ls], it now does it only in a reduced interval [L(k+1), Ls], where Lk represents the higher length of the code beginning kept in memory.

The number of free lengths is now N(fl) = Ls – Lk. To give more flexibility to the method, three levels of freedom have been defined :

a) fixed length, which corresponds to the lengths where the number of codewords is fixed and used for all the next values of Ls ;

b) variance length, which is the set of lengths where a slight freedom (± 1) is tolerated ;

c) free length, which corresponds to the set of lengths where all possible numbers of codewords are tested.

As in Ls optimization, the rest of the length distribution, or tail lengths, corresponds to lengths above the Ls limit.

The introduction of the variance length part results directly from practical observations, when it appeared during simulations that, in order to improve the results, it was necessary to establish a slight freedom with a variance of ± 1 on the number of codewords found at the higher fixed length Lk. Higher variance levels were tried, and the same results were obtained, even with other high numbers of codeword sources. It seems possible to improve the performance, but it requires more computation time. Several methods can be used to reach this goal. A first one may consist in increasing the size of the free length set. Another way to reach this goal would be to increase the number of lengths in the variance-length set, for instance by putting not only one but two lengths, Lk and L(k-1), in the variance length set with an incertitude of ± 1 (as there are few variations of the number of codewords at these lengths, this method should give the same results as the previous one while being faster).

The present method, also called BestAllure optimization, allows very noticeable gains of time because it decreases the time factor between each Ls (approximately from 30 to 2). This is principally due to the fixed size of the free length when using Ls optimization. In practice, it is especially interesting for high

number of symbols source, where exhaustive methods are impossible and where existing one, like those proposed by Buttigieg, are untractable.